Appendix-G: *Storage Class Specifiers*

The following are the storage class specifiers in C++. Only the highlighted will be describe in detail.

auto	**static**
extern	**typedef**
register	

auto

Syntax
[auto] <data-definition> ;

Description
Use the auto modifier to define a local variable as having a local lifetime.
This is the default for local variables and is rarely used.

extern

Syntax
extern <data definition> ;
[extern] <function prototype> ;

Description
Use the extern modifier to indicate that the actual storage and initial value of a variable, or body of a function, is defined in a separate source code module. Functions declared with extern are visible throughout all source files in a program, unless you redefine the function as static.
The keyword extern is optional for a function prototype.
Use extern "c" to prevent function names from being mangled in C++ programs.

register

Syntax
register <data definition> ;

Description
Use the register storage class specifier to store the variable being declared in a CPU register (if possible), to optimize access and reduce code.
Items declared with the register have a global lifetime.

static

Syntax
static <data definition> ;
static <function definition> ;

Description
Use the static storage class specifier with a local variable to preserve the last value between successive calls to that function. A static variable acts like a local variable but has the lifetime of an external variable.

typedef

Syntax
typedef <type definition> <identifier> ;

Description
Use the typedef keyword to assign the symbol name <identifier> to the data type definition <type definition>.